(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 05346922 A

(43) Date of publication of application: 27 . 12 . 93

(51) Int. Cl G06F 15/20
G06F 9/06

(21) Application number: 04153946

(22) Date of filing: 15 . 06 . 92

(71) Applicant: PFU LTD

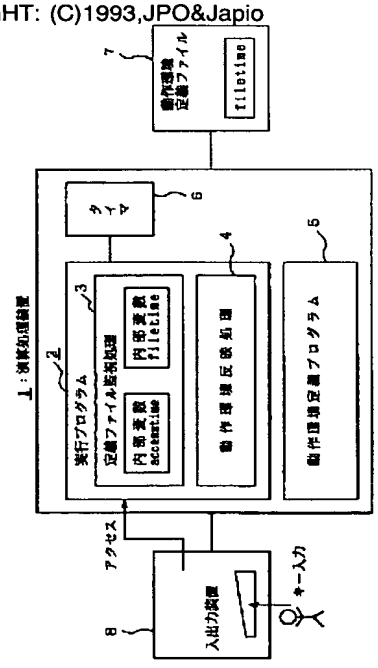
(72) Inventor: OKUDAIRA YUMIKO
SHINDO TOKUMI
WATARI AKIKO
YAMADA TOSHIAKI
KOIKE TADASHI

(54) DYNAMIC REFLECTING METHOD FOR OPERATION ENVIRONMENT

(57) Abstract:

PURPOSE: To reduce the load and to speedily and dynamically reflect operation environment on an execution program by monitoring access and reflecting the operation environment on the execution program only when the access interval is longer than a specific time and an operation environment definition file is updated as to the dynamic reflecting method which dynamically reflects the operation environment on the execution program.

CONSTITUTION: This method is equipped with a definition file monitoring process 3 for monitoring whether or not the interval of access to the execution program exceeds the specific time and an operation environment definition file 7 wherein the operation environment is set; and the access is monitored through the definition file monitoring process 3 and if it is made evident that the operation environment definition file 7 is updated by referring to the operation environment definition file 7 when a state wherein access is attained longer than the specific time after a last access time is detected, the updated operation environment is taken out of the operation environment definition file 7 to update the operation environment of the execution program, thereby performing processes under the new operation environment.



Japanese Unexamined Patent Application Publication

No. 5-346922

[Title of the Invention] DYNAMIC REFLECTING METHOD OF OPERATING ENVIRONMENT

[0019] Fig. 1 is a configuration diagram of an embodiment of the present invention. In Fig. 1 a processing unit 1 carries out various processes. The processing unit 1 comprises an execution program 2 performing Kana-Kanji conversion and the like, an operating environment defining program 5 setting an operating environment in an operating environment definition file 7, a timer 6 and the like.

[0020] The execution program 2 is a program conducting various kinds of processing such as Kana-Kanji conversion, and comprises, in this embodiment, a definition file monitoring and an operating environment reflecting 4.

[0021] The definition file monitoring 3 monitors access from the input/output unit 8 to the execution program 2 to see whether or not the interval between accesses has exceeded a prescribed period of time. This access monitoring is to determine the necessity of reflecting the operating environment onto the execution program 2, when the execution program 2 is accessed, on condition that the difference between the current time fetched with reference to the

timer 6 and the time fetched from the internal variable "accesstime" storing the last access time is larger than a certain value, and the time fetched from the internal variable "filetime" storing the last update time of the operating environment definition file 7 and the time fetched from the variable "filetime" of the operating environment definition file 7 are not equal to each other, and the operating environment has been updated.

[0022] The internal variable "accesstime" is a variable storing the last access time. The internal variable "filetime" is a variable storing the time upon the last update of the operating environment definition file 7.

[0023] The operating environment reflecting 4 is to set the operating environment fetched from the operating environment definition file 7 in the execution program 2, and causes execution of processing in the thus set new operating environment.

[0024] The operating environment definition program 5 defines (sets, updates) an operating environment specified by the operator from the input-output unit 8 or the like in the operating environment definition file 7. In this definition, the variable "filetime" is previously updated time (see Fig. 3).

[0025] The timer 6 is to count the current time. The operating environment definition file 7 is a file storing definitions of the

current environments, and has a variable "filetime" storing the time when making a definition (setting, updating). The execution file monitoring 3 compares the time of the internal variable "filetime" and the time of this variable "filetime", and they differ from each other. If different, the operating environment is determined to have been updated.

[0026] The input/output 8 conducts various operations of input and output, comprising, for example, a keyboard and a display. The operations with the configuration shown in Fig. 1 will now be described in detail in accordance with the sequence shown in the flowchart of Fig. 2.

[0027] In Fig. 2, initial values of the internal variable "filetime" and the internal variable "accesstime" are set in S1. Initial values are set with reference to the operating environment definition file 7 and from the timer to the current time. For example, initialization is conducted as follows:

Initial variable filetime ← f0 (time f0 set for the variable filetime of the operating environment definition file 7), and

Internal variable accesstime ← a0 (current time of timer 6)

[0028] S2 is a step of waiting for an access from the operator. S3 is a step of determining whether or not an operator access has occurred. This is accomplished by the operator from the keyboard

which is the input/output unit 8 shown in Fig. 1 by, for example, entering a character string to be subjected to Kana-Kanji conversion to determine whether or not an access to the execution program 2 has occurred. In the case of YES, the process proceeds to S4 and the subsequent steps. In the case of NO, suggesting occurrence of no access, the process returns to S2.

[0029] S4 is a step of acquiring the current time a1. This step comprises acquiring the current time a1 from the timer 6. In S5, it is determined whether or not a certain period of time has elapsed from the time a0 stored in the internal variable accesstime to the current time a1 acquired in S4.

This is determined by:

a1 - a0 > certain period of time

In the case of YES, suggesting that a certain period of time has elapsed from the last access to the current time, the process goes to S6. In the case of NO, on the other hand, revealing that no certain period of time has elapsed from the last access to the current time, reflecting the operating environment in the execution program is discontinued. The internal accesstime is updated to the current time (internal variable accesstime ← a1) in S10, and processing is carried out in S12, to return to S2.

[0030] S6 is a step of acquiring the definition file time f1 of the

operating environment definition file 7. This is accomplished by fetching the time f1 stored in the variable filetime of the operating environment definition file 7 shown in Fig. 1.

[0031] In S7, it is determined whether or not the definition file time f1 of the operating environment definition file 7 has been updated. This is the determination of f1 ≠ f0. More specifically, this is to compare the time f0 of the variable filetime of the operating environment definition file 7 when fetching an operating environment from the operating environment definition file 7 last time and reflecting the same in the execution program 2, and the time f1 of the variable filetime of the current operating environment definition file 7, and to check if these values are not equal (updated). If YES, which suggests update of the operating environment definition file 7, the process transfers to S8. If NO, on the other hand, suggesting that the operating environment definition file 7 is not updated, the interval accesstime is updated into a current time (internal variable accesstime ← a1) in S11, and processing is executed in S12, returning to S2.

[0032] In S8, the internal variable accesstime is updated into the current time, and the internal variable filetime is updated into the current time. In other words, updating is accomplished as follows:

Internal variable accesstime ← a1 (current time)

Internal variable filetime ← f1 (updated time f1 of the operating environment definition file 7).

These steps S4 to S8, S10 and S11 are carried out by the definition file monitoring 3 shown in Fig. 1.

[0033] In S9, the execution program 2 reflects the operating environment definition file 7, i.e., fetches an operating environment after updating environment from the operating environment definition file 7, sets the same into an executable state, and executes processing under this operating environment in S12, for example, carries out Kana-Kanji conversion. S9 is accomplished by the operating refection4.

[0034] As a result of the above, only when the interval between accesses from the input/output unit 8 to the execution program 2 is over a certain period of time (YES in S5), and the operating environment definition file 7 has been updated (YES in S7), the updated operating environment is fetched from the operating environment definition file 7; the fetched operating environment is reflected onto the execution program; and processing is executed on the basis of this updated operating environment. Accordingly, when the execution program 2 is frequently accessed by the operator via the input/output unit 8 such as the keyboard, the load is alleviated by not referring to the operating environment file 7. Since a

certain period of time is necessary for updating the operating environment definition file 7, only when an access is made after the lapse of the certain period of time expected to be necessary for updating, it is checked up whether or not the operating environment has been updated with reference to the operating environment definition file 7. When updated, the operating environment is quickly reflectedin the execution program.

FIG. 1

CONFIGURATION DIAGRAM OF FIRST EMBODIMENT OF THE INVENTION

1: PROCESSING UNIT

2: EXECUTION PROGRAM

3: DEFINITION FILE MONITORING

INTERNAL VARIABLE accesstime

4: OPERATING ENVIRONMENT REFLECTION

5: OPERATING ENVIRONMENT DEFINITION PROGRAM

6: TIMER

7: OPERATING ENVIRONMENT DEFINITION FILE

8: INPUT/OUTPUT UNIT

(1) ACCESS

(2) KEY-INPUT

FIG. 2

REFLECTING FLOWCHART OF OPERATING ENVIRONMENT DEFINITION FILE OF THE INVENTION

S1: INITIALIZATION OF INTERNAL VARIABLES filetime AND accesstime

SET WITH REFERENCE TO filetime IN DEFINITION FILE;

SET WITH REFERENCE TO CURRENT TIME FROM TIMER;

filetime ← f0

accesstime ← a0

S2: WAITING FOR ACCESS FROM OPERATOR

S3: HAS OPERATOR ACCESS OCCURRED?

S4: ACQUIRE CURRENT TIME (a1)

S5: HAS CERTAIN TIME ELAPSED FROM accesstime?

(a1 - a0 . certain time)

S6: ACQUIRE DEFINITION FILE TIME f1

S7: HAS DEFINITION FILE TIME BEEN UPDATED? (f1 ≠ f0)

S8: UPDATE accesstime (accesstime ← a1)

update filetime (filetime ← f1)

S9: EXECUTION PROGRAM REFLECTS DEFINITION FILE

S10: UPDATE accesstime (accesstime ← a1)

S11: UPDATE accesstime (accesstime ← a1)

S12: EXECUTE PROCESSING

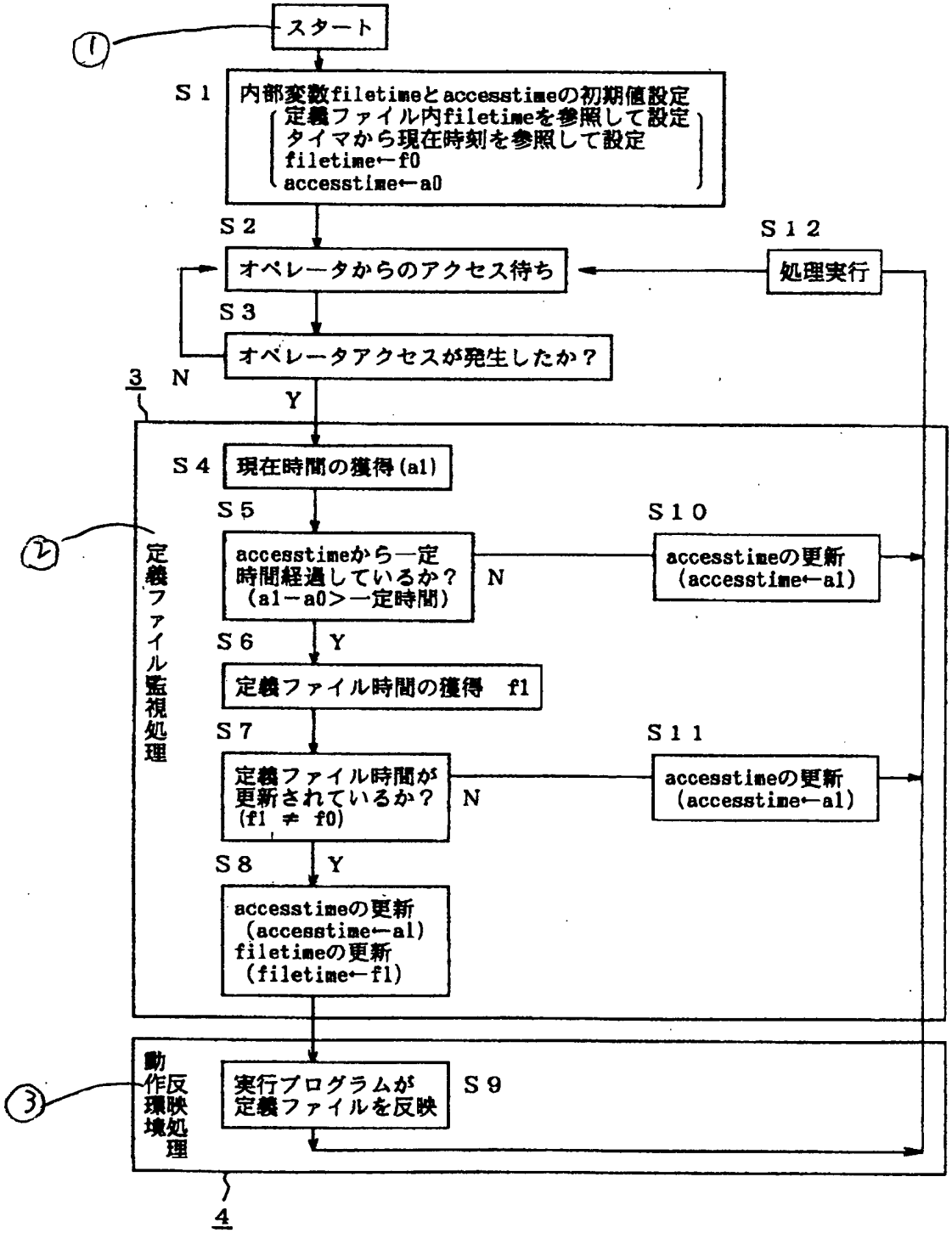
(1) START

(2) DEFINITION FILE MONITORING

(3) OPERATING ENVIRONMENT REFLECTION

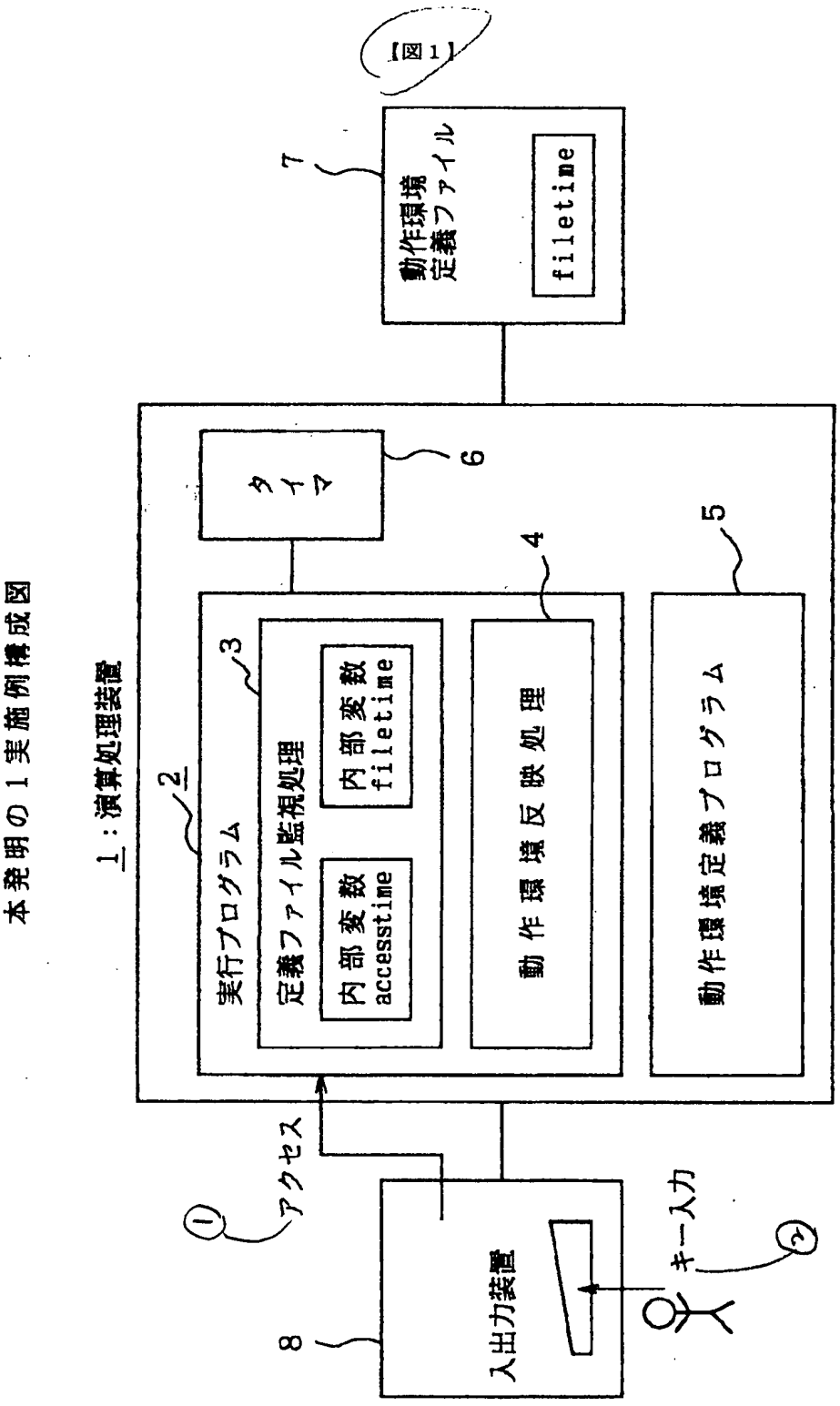
【図2】

本発明の動作環境定義ファイルの反映フローチャート



【図１】

本発明の１実施例構成図



(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平５－３４６９２２

(43)公開日　平成5年(1993)12月27日

(51)Int.Cl.[5]		識別記号	庁内整理番号	F I	技術表示箇所
G06F	15/20	586 G	7343－5L		
	9/06	410 B	9367－5B		

審査請求　有　請求項の数２　(全１２頁)

(21)出願番号　特願平4-153946

(22)出願日　平成4年(1992)6月15日

(71)出願人　000136136
株式会社ピーエフユー
石川県河北郡宇ノ気町字宇野気ヌ98番地の2

(72)発明者　奥平　由美子
神奈川県大和市深見西四丁目2番49号　株式会社ピーエフユー大和工場内

(72)発明者　進藤　徳海
神奈川県大和市深見西四丁目2番49号　株式会社ピーエフユー大和工場内

(72)発明者　渡利　亜紀子
神奈川県大和市深見西四丁目2番49号　株式会社ピーエフユー大和工場内

(74)代理人　弁理士　岡田　守弘

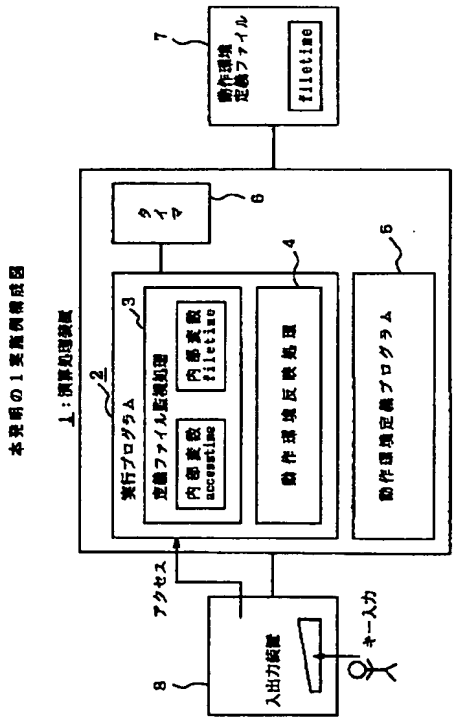
最終頁に続く

(54)【発明の名称】動作環境の動的反映方法

(57)【要約】

【目的】　本発明は、動作環境を動的に実行プログラムに反映する動的反映方法に関し、アクセスを監視してアクセス間隔が所定時間以上であって、かつ動作環境定義ファイルが更新されているときにのみ動作環境を実行プログラムに反映し、負荷を少なくしてかつ迅速に動作環境を動的に実行プログラムへの反映を可能にすることを目的とする。

【構成】　実行プログラムへのアクセスの間隔が所定時間以上経過したかを監視する定義ファイル監視処理３と、動作環境を設定する動作環境定義ファイル７とを備え、定義ファイル監視処理３がアクセスを監視し、アクセスが前回のアクセス時間から所定時間以上経過していることを検出したときに、動作環境定義ファイル７を参照して更新されていると判明したときに当該動作環境定義ファイル７から更新後の動作環境を取り出し、実行プログラムの動作環境を更新し、新たな動作環境のもとで処理を実行するように構成する。



1

【特許請求の範囲】

【請求項１】動作環境を動的に実行プログラムに反映する動的反映方法において、

実行プログラムへのアクセスの間隔が所定時間以上経過したかを監視する定義ファイル監視処理（３）と、

動作環境を設定する動作環境定義ファイル（７）とを備え、

上記定義ファイル監視処理（３）がアクセスを監視し、アクセスが前回のアクセス時間から所定時間以上経過していることを検出したときに、上記動作環境定義ファイル（７）を参照して更新されていると判明したときに当該動作環境定義ファイル（７）から更新後の動作環境を取り出し、実行プログラムの動作環境を更新し、新たな動作環境のもとで処理を実行するように構成したことを特徴とする動作環境の動的判定方法。

【請求項２】上記定義ファイル監視処理（３）が前回のアクセス時間を記憶する内部変数accesstimeおよび上記動作環境定義ファイル（７）の前回の更新時間を記憶する内部変数filetimeを持つと共に、当該動作環境定義ファイル（７）に更新時間を記憶する変数filetimeを持ち、

この動作環境定義ファイル（７）を更新したときに変数filetimeを更新しておき、

定義ファイル監視処理（３）がアクセスを監視し、現在のアクセス時間と上記内部変数accesstimeの時間とを比較および更新し、所定時間以上経過していると判明したときに、上記動作環境定義ファイル（７）の変数filetimeを参照して内部変数filetimeと異なると判明したときに当該動作環境定義ファイル（７）から更新後の動作環境を取り出して実行プログラムの動作環境に反映および内部変数filetimeを更新するように構成したことを特徴とする請求項１記載の動作環境の動的判定方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、動作環境を動的に実行プログラムに反映する動的反映方法に関するものである。

【０００２】

【従来の技術】従来、かな漢字変換システムなどは、動作環境であるキーバインドや変換方法などをオペレータが任意に設定し、操作し易いようにキーボード中の任意のキーに文字、記号を割り当てたり、変換した結果、得られる漢字の範囲（ＪＩＳ第１水準、ＪＩＳ第２水準など）を変更したり、更に専門分野に応じて辞書を切り替えたりしている。

【０００３】また、動作環境の設定は、プログラムの動作中も行える。この場合には、更新した動作環境をかな漢字変換システムに反映させるためには、プログラムを一旦停止させ、再起動して動作環境を取り込む。

【０００４】このように動作環境を更新したときにプログラムを一旦停止し、再起動させて反映することは不便であるので、プログラムの動作中に動作環境定義ファイルが更新されたときに、プログラムを停止することなく反映することが望まれ、これを達成するために以下のような方法が考えられる。

2

【０００５】（１）　第１の方法は、オペレータからのアクセス毎に動作環境定義ファイルが更新されているか否かを常にチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う。

【０００６】（２）　第２の方法は、一定周期に割込で動作環境定義ファイルが更新されているか否かをチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う。

【０００７】（３）　第３の方法は、動作環境定義ファイルをチェックした時間を記憶しておき、オペレータからのアクセス毎にこの記憶した時間を参照して前回から一定時間以上経過していたら動作環境定義ファイルが更新されているか否かをチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う共に、チェックした時間を記憶する。

【０００８】

【発明が解決しようとする課題】従来は、プログラムの動作中に動作環境定義ファイルが更新されたときに、プログラムにその更新後の動作環境を反映するために、上記（１）、（２）、（３）の方法が考えられるが、これらの方法について、図５、図６、図７に示すように、時間３１で動作環境定義ファイルの更新が発生した場合、それぞれ以下の問題がある。

【０００９】上記（１）の第１の方法は、図５に示すように、動作環境定義ファイルの更新をすぐに補足し、時間３２で動作環境をプログラムに反映できるが、１３回の動作環境定義ファイルへのアクセスが必要となり、負荷が大きいという問題がある。

【００１０】上記（２）の第２の方法は、図６に示すように、一定時間５で割り込みを発生させた場合、時間３５で動作環境をプログラムに反映できるが、８回の動作環境定義ファイルへのアクセスが必要となり、アクセス回数が減って負荷は少し減少するが、プログラムへの反映が時間３５となって遅れてしまうという問題がある。

【００１１】上記（３）の第３の方法は、図７に示すように、動作環境定義ファイルをアクセスしてチェックしてから次のチェックまでの時間差分５とした場合、時間３２で動作環境をプログラムに反映できるが、７回の動作環境定義ファイルへのアクセスが必要となり、アクセス回数が減って負荷は少し減少するが、まだ７回もの動作環境定義ファイルへのアクセスが行われてしまう問題がある。

【００１２】本発明は、これらの問題を解決するため、アクセスを監視してアクセス間隔が所定時間以上であって、かつ動作環境定義ファイルが更新されているときに

のみ動作環境を実行プログラムに反映し、負荷を少なくしてかつ迅速に動作環境を動的に実行プログラムへの反映を可能にすることを目的としている。

【００１３】

【課題を解決するための手段】図１を参照して課題を解決するための手段を説明する。図１において、定義ファイル監視処理３は、アクセスの間隔が所定時間以上経過したか監視するものであって、前回のアクセス時間を記憶する内部変数accesstimeおよび動作環境定義ファイル７の前回の更新時間を記憶する内部変数filetimeを持つものである。

【００１４】動作環境定義ファイル７は、動作環境を保持するものであって、更新時間を記憶する変数filetimeを内部に持つものである。

【００１５】

【作用】本発明は、図１に示すように、定義ファイル監視処理３がアクセスを監視し、アクセスが前回のアクセス時間から所定時間以上経過していることを検出したときに、動作環境定義ファイル７を参照して更新されていると判明したときに当該動作環境定義ファイル７から更新後の動作環境を取り出し、実行プログラムの動作環境を更新し、新たな動作環境のもとで処理を実行するようにしている。

【００１６】また、定義ファイル監視処理３がアクセスを監視し、現在のアクセス時間と内部変数accesstimeの時間とを比較および更新し、所定時間以上経過していると判明したときに、動作環境定義ファイル７の変数filetimeを参照して内部変数filetimeと異なると判明したときに当該動作環境定義ファイル７から更新後の動作環境を取り出して実行プログラムに反映すると共に内部変数filetimeを更新し、新たな動作環境のもとで処理を実行するようにしている。

【００１７】従って、アクセスを監視してアクセス間隔が所定時間以上であって、かつ動作環境定義ファイル７が更新されているときにのみ動作環境を実行プログラムに反映することにより、負荷を少なくして動作環境を動的に実行プログラムに迅速に反映することが可能となる。

【００１８】

【実施例】次に、図１から図４を用いて本発明の実施例の構成および動作を順次詳細に説明する。

【００１９】図１は、本発明の１実施例構成図を示す。図１において、演算処理装置１は、各種演算処理を行うものである。この演算処理装置１は、かな漢字変換などを行う実行プログラム２、動作環境定義ファイル７に動作環境を設定する動作環境定義プログラム５、およびタイマ６などから構成されるものである。

【００２０】実行プログラム２は、各種処理を行うプログラムであって、例えばかな漢字変換を行うプログラムであり、ここでは、定義ファイル監視処理３および動作環境反映処理４などから構成されるものである。

【００２１】定義ファイル監視処理３は、入出力装置８などからの実行プログラム２に対するアクセスを監視し、アクセスの間隔が所定時間以上経過したか監視するものである。このアクセスの監視は、アクセスがあったときに、タイマ６を参照して取り出した現時間と、前回のアクセス時間を記憶する内部変数accesstimeから取り出した時間との差が一定以上であって、かつ動作環境定義ファイル７の前回の更新時間を記憶する内部変数filetimeから取り出した時間と、動作環境定義ファイル７の変数filetimeから取り出した時間とを比較し、等しくなくて動作環境が更新されており、動作環境を実行プログラム２に反映する必要があるか判定するものである。

【００２２】内部変数accesstimeは、前回のアクセス時間を記憶する変数である。内部変数filetimeは、前回の動作環境定義ファイル７を更新したときの時間を記憶する変数である。

【００２３】動作環境反映処理４は、動作環境定義ファイル７から取り出した動作環境を実行プログラム２に設定し、この設定後の新たな動作環境で処理を実行させるものである。

【００２４】動作環境定義プログラム５は、入出力装置８などからオペレータによって指示された動作環境を、動作環境定義ファイル７に定義（設定、更新）するものである。この際、変数filetimeを更新した時間に設定しておく(図３参照)。

【００２５】タイマ６は、現時間を計数するものである。動作環境定義ファイル７は、動作環境の定義を格納するファイルであって、定義（設定、更新）したときの時間を記憶する変数filetimeを持つものである。実行ファイル監視処理３が内部変数filetimeの時間とこの変数filetimeの時間とを比較し、異なるときに動作環境が更新されていると判定する。

【００２６】入出力装置８は、各種入出力を行うものであって、例えばキーボードやディスプレイである。次に、図２のフローチャートに示す順序に従い、図１の構成の動作を詳細に説明する。

【００２７】図２において、Ｓ１は、内部変数filetimeと、内部変数accesstimeの初期値設定する。初期値は、動作環境定義ファイル７を参照して設定、およびタイマから現在時間を参照して設定する。例えば

内部変数filetime←ｆ０（動作環境定義ファイル７の変数filetimeに設定されている時間ｆ０）

内部変数accesstime←ａ０（タイマ６の現時間ａ０）

と初期設定する。

【００２８】Ｓ２は、オペレータからのアクセス待ちする。Ｓ３は、オペレータアクセスが発生したか判別する。これは、図１の入出力装置８であるキーボードからオペレータが例えばかな漢字変換する文字列を入力して実行プログラム２に対するアクセスが発生したか判別す

る。ＹＥＳの場合には、Ｓ４以降に進む。一方、ＮＯの場合には、アクセスが発生しないのでＳ２に戻る。

【００２９】Ｓ４は、現在時間ａ１の獲得を行う。これは、現在の時間ａ１をタイマ６から獲得する。Ｓ５は、内部変数accesstimeに記憶している時間ａ０から、Ｓ４で獲得した現在の時間ａ１までの間に一定時間が経過しているか判別する。これは、

ａ１－ａ０＞一定時間

か判別する。ＹＥＳの場合には、前回にアクセスした時間から現在の時間までの間に一定時間が経過していたので、Ｓ６に進む。一方、ＮＯの場合には、前回にアクセスした時間から現在の時間までの間に一定時間が経過していないので、動作環境を実行プログラムに反映することを止め、Ｓ１０で内部accesstimeを現時間に更新(内部変数accesstime←ａ１）し、Ｓ１２で処理を実行し、Ｓ２に戻る。

【００３０】Ｓ６は、動作環境定義ファイル７の定義ファイル時間ｆ１の獲得を行う。これは、図１の動作環境定義ファイル７の変数filetimeに記憶されている時間ｆ１を取り出す。

【００３１】Ｓ７は、動作環境定義ファイル７の定義ファイル時間ｆ１が更新されているか判別する。これは、ｆ１≠ｆ０か判別する。即ち前回に動作環境定義ファイル７から動作環境を取り出して実行プログラム２に反映したときの当該動作環境定義ファイル７の変数filetimeの時間ｆ０と、現在の動作環境定義ファイル７の変数filetimeの時間ｆ１とを比較し、等しくない（更新されている）か判別する。ＹＥＳの場合には、動作環境定義ファイル７が更新されていると判明したので、Ｓ８に進む。一方、ＮＯの場合には、動作環境定義ファイル７が更新されていないと判明したので、Ｓ１１で内部accesstimeを現時間に更新(内部変数accesstime←ａ１）し、Ｓ１２で処理を実行し、Ｓ２に戻る。

【００３２】Ｓ８は、内部変数accesstimeを現時間に更新、および内部変数filetimeを現時間に更新する。即ち、

内部変数accesstime←ａ１（現時間）

内部変数filetime←ｆ１（動作環境定義ファイル７の更新時間ｆ１）

とする。これらＳ４からＳ８、Ｓ１０、Ｓ１１は図１の定義ファイル監視処理３が行う。

【００３３】Ｓ９は、実行プログラム２が動作環境定義ファイル７を反映、即ち動作環境定義ファイル７から更新後の動作環境を取り出し、実行し得る状態に設定し、Ｓ１２でこの動作環境のもとで処理を実行、例えばかな漢字変換を行う。ここで、Ｓ９は動作反映処理４が行う。

【００３４】以上によって、入出力装置８から実行プログラム２に対するアクセス間隔が一定時間以上であって（Ｓ５のＹＥＳ）、かつ動作環境定義ファイル７の動作環境が更新されていた場合（Ｓ７のＹＥＳ）にのみ、動作環境定義ファイル７から更新された動作環境を取り出し、実行プログラム２に反映し、この更新された動作環境をもとに処理を実行する。これらにより、入出力装置８である例えばキーボードなどからオペレータにより実行プログラム２が頻繁にアクセスされているときは動作環境ファイル７を参照しないようにして負荷を軽減し、一方、動作環境定義ファイル７を更新するには一定時間必要であるので、この更新に要すると予想される一定時間を経過した後にアクセスがあった場合にのみ動作環境定義ファイル７を参照して動作環境が更新されているか否かをチェックし、更新されている場合に動作環境を実行プログラムに迅速に反映する。

【００３５】図３は、本発明の動作環境定義ファイルの変更フローチャートを示す。これは、オペレータからの指示に従って、図１の動作環境定義ファイル７を更新するときの手順である。

【００３６】図３において、Ｓ２１は、オペレータが定義変更コマンドを押下する。これは、図１の入出力装置８であるキーボード上でオペレータが特定の定義変更コマンドのキーを押下し、図１の動作環境定義プログラム５を起動する。

【００３７】Ｓ２２は、選択可のパターンを表示する。これは、Ｓ２１の定義変更コマンドの押下に対応して、例えばかな漢字変換の動作環境として、現在、あるキーを押下したとき、“．”と表示されるが他の選択可のパターン“。”　“、”などを表示する。

【００３８】Ｓ２３は、選択したか判別する。ＹＥＳの場合には、Ｓ２４に進む。ＮＯの場合には、Ｓ２２で選択可のパターンを表示し、選択するまで待機する。Ｓ２４は、動作環境定義ファイル７を新たな動作環境に更新する。

【００３９】Ｓ２５は、動作環境定義ファイル７の変数filetimeを現在の時間に更新する。以上によって、オペレータが定義変更コマンドを押下して実行プログラム２の動作環境を保存する動作環境定義ファイル７の動作環境を更新する。この動作環境定義ファイル７の動作環境を更新した状態では、実行プログラム２の動作環境は更新されず、次回の起動を待たなければ更新されない。実行プログラム２の動作中に動的に動作環境を更新するには、既述した図２の手順によって動的に更新する。

【００４０】図４は、本発明の具体例説明図を示す。これは、図３のフローチャートに従って動作環境定義ファイル７が更新されたときに、図２のフローチャートに従い当該動作環境定義ファイル７から動作環境を動的に実行プログラム２に反映するときの具体例である。ここで

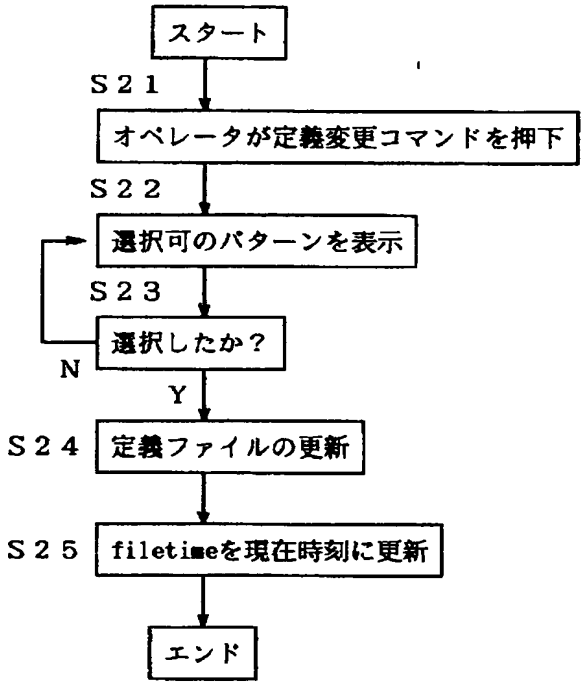
・オペレータのアクセスタイミングは、図１でオペレータが入出力装置８であるキーボードからキー入力して“かな”を入力して実行プログラム２をアクセスするタイミングである。

【００４１】・時間軸は、時間０から時間３５をここでは刻む。
・accesstimeの更新点と、時間差分は、定義ファイル監視処理３が持つ内部変数accesstimeの更新点と、時間差分を示す。
【００４２】・実行プログラムの動作は、図１の実行ログラム２の動作を示す。
次に、動作を説明する。
（１）　時間０：オペレータのアクセスタイミングであるので、実行プログラム２の定義ファイル監視処理３がファイルチェックする。この初期状態では、動作環境が更新されていないので、動作環境を実行プログラム２に反映しない（図２のＳ１）。
【００４３】（２）　時間２：オペレータのアクセスタイミングであるので、実行プログラム２の定義ファイル監視処理３がアクセスタイムの経過時間について一定時間５を越えていないので（図２のＳ５のＮＯ）、内部変数accesstimeを現時間に更新し、動作環境定義ファイル７を参照しない。
【００４４】（３）　同様に、時間５、時間７、時間９、時間１１についても一定時間４を越えていないので（図２のＳ５のＮＯ）、内部変数accesstimeを現時間にそれぞれ更新し、動作環境定義ファイル７を参照しない。
【００４５】（４）　時間１６：実行プログラム２の定義ファイル監視処理３がアクセスタイムの経過時間について一定時間４を越えているので（図２のＳ５のＹＥＳ）、動作環境定義ファイル７をチェックするが動作環境が更新されていないので（図２のＳ７のＮＯ）、動作環境の反映を行わない。
【００４６】（５）　時間１８、時間２０、時間２２、時間２５、時間２７についても一定時間４を越えていないので（図２のＳ５のＮＯ）、内部変数accesstimeを現時間にそれぞれ更新し、動作環境定義ファイル７を参照しない。
【００４７】（６）　時間３２：実行プログラム２の定義ファイル監視処理３がアクセスタイムの経過時間について一定時間４を越えているので（図２のＳ５のＹＥＳ）、動作環境定義ファイル７をチェックすると動作環境が、動作環境定義プログラム５によって時間３１のときに更新されていたので（図２のＳ７のＹＥＳ）、動作環境を取り出して実行プログラム２に反映する。
【００４８】以上の手順により、ファイルチェック（動作環境定義ファイル７を参照して動作環境が更新されているかのチェック）を行う回数がここでは、３回で済み、しかも動作環境定義ファイル７をオペレータが更新操作を行うと、オペレータアクセスタイミングがここでは一定時間４を越えてしまう特質を利用し、迅速に動作環境定義ファイル７の更新を検出し、更新後の動作環境を動的に迅速に実行プログラム２に反映し、反映した後の動作環境で処理を実行することが可能となる。
【００４９】尚、図１から図４を用いて説明した構成をかな漢字変換システムに使用し、動作環境を更新する例としては以下の場合がある。
（１）　変換辞書の動作環境を更新する場合：個別辞書である、人名辞書、地名辞書、専門用語辞書などを必要に応じて追加したり、削除したり、切り替えたりを行う。
【００５０】（２）　かな漢字変換する文字列の動作環境を更新する場合：
・複文節解析
複文節変換時に意味優先か学習優先かを選択する。
【００５１】・複合語
変換時に複合語優先か否かを選択する。
・数詞変換
数詞変換を行うか否かを選択する。
【００５２】・ＪＩＳ漢字
変換対象文字を常用漢字のみにするか、ＪＩＳ第２水準以外の文字にするか、全ての文字にするかを選択する。
【００５３】（３）　読み文字の入力の動作環境を更新する場合：
・カンマの処理
カンマのキーを打鍵した時、カンマを表示するか、読点を表示するかを選択する。
【００５４】・ピリオドの処理
ピリオドのキーを打鍵した時、ピリオドを表示するか、句点を表示するかを選択する。
【００５５】（４）　変換する時のスタイルの動作環境を更新する場合：
・次変換時カーソル位置
変換する時のカーソルの位置を選択する。
【００５６】・変換文字列の属性の動作環境を更新する場合：変換する時の文字列の属性（下線を引くか、反転させるかなど）を選択する。
【００５７】・候補群選択画面の動作環境を更新する場合：候補群を出す時の画面スタイルを選択する。
（５）　キーバインドの動作環境を更新する場合：
・変換のためのキーをキーボード上のどのキーに割り当てるかなどを変更する。
【００５８】
【発明の効果】以上説明したように、本発明によれば、アクセスを監視してアクセス間隔が所定時間以上であって、かつ動作環境定義ファイル７が更新されているときにのみ動作環境を実行プログラムに反映する構成を採用しているため、負荷を少なくして動作環境を動的に実行プログラムに迅速に反映することができる。
【図面の簡単な説明】
【図１】本発明の１実施例構成図である。
【図２】本発明の動作環境定義ファイルの反映フローチャートである。

【図３】本発明の動作環境定義ファイルの変更フローチャートである。
【図４】本発明の具体例説明図である。
【図５】従来技術の説明図（その１）である。
【図６】従来技術の説明図（その２）である。
【図７】従来技術の説明図（その３）である。
【符号の説明】
１：演算処理装置
２：実行プログラム
３：定義ファイル監視処理
４：動作環境反映処理
５：動作環境定義プログラム
６：タイマ
７：動作環境定義ファイル
８：入出力装置

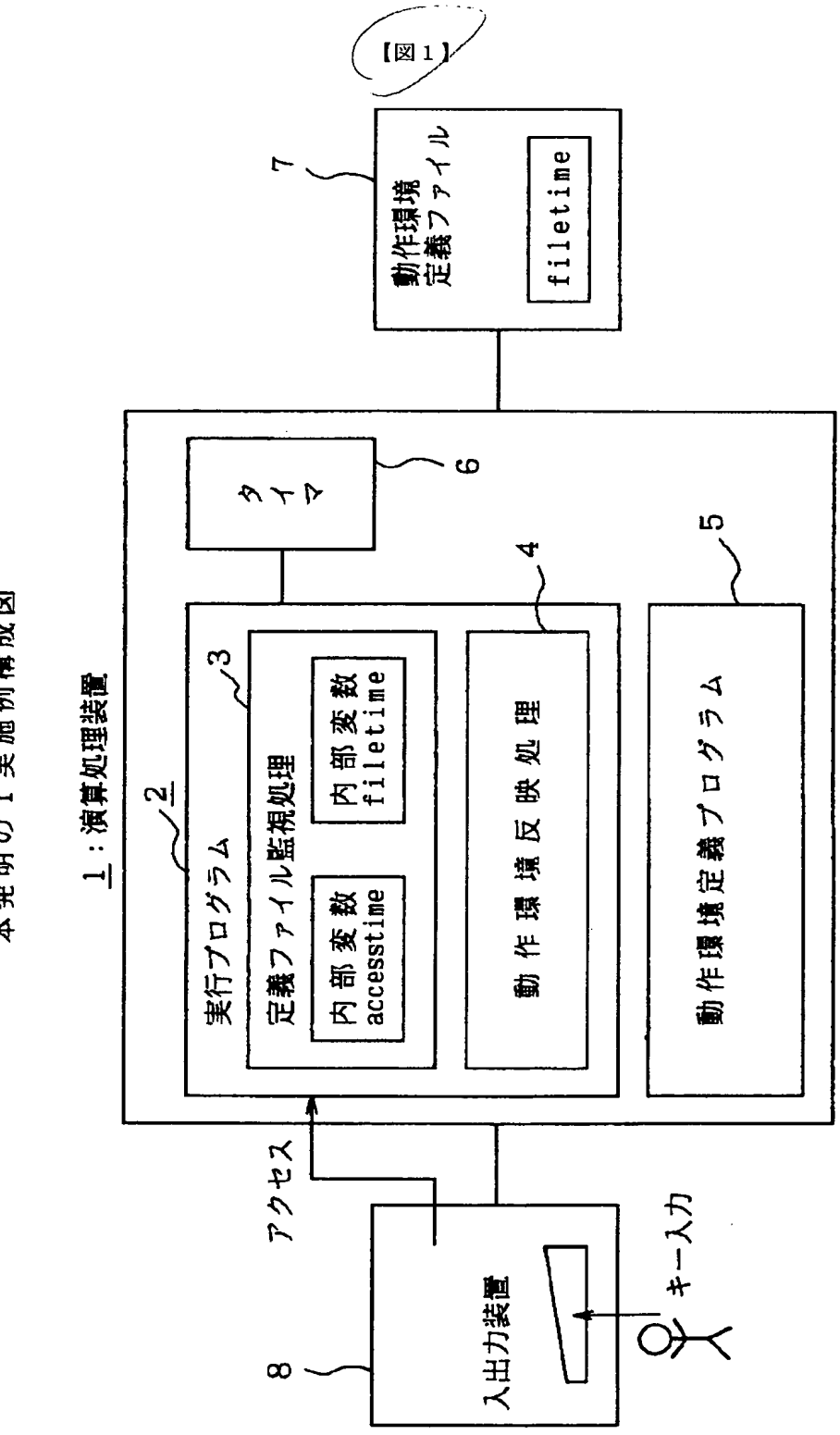
【図３】

本発明の動作環境定義ファイルの変更フローチャート



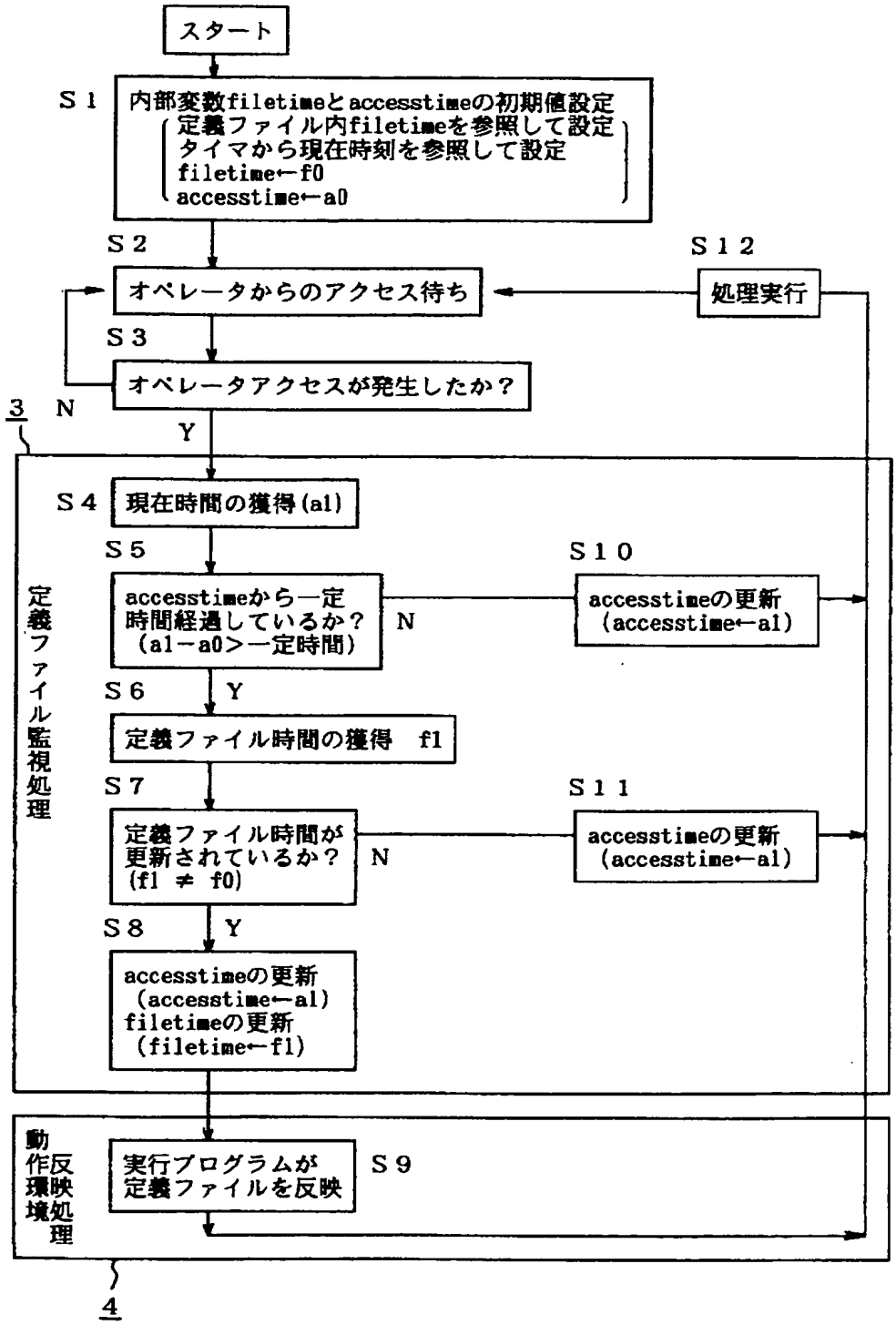
【図１】

本発明の１実施例構成図



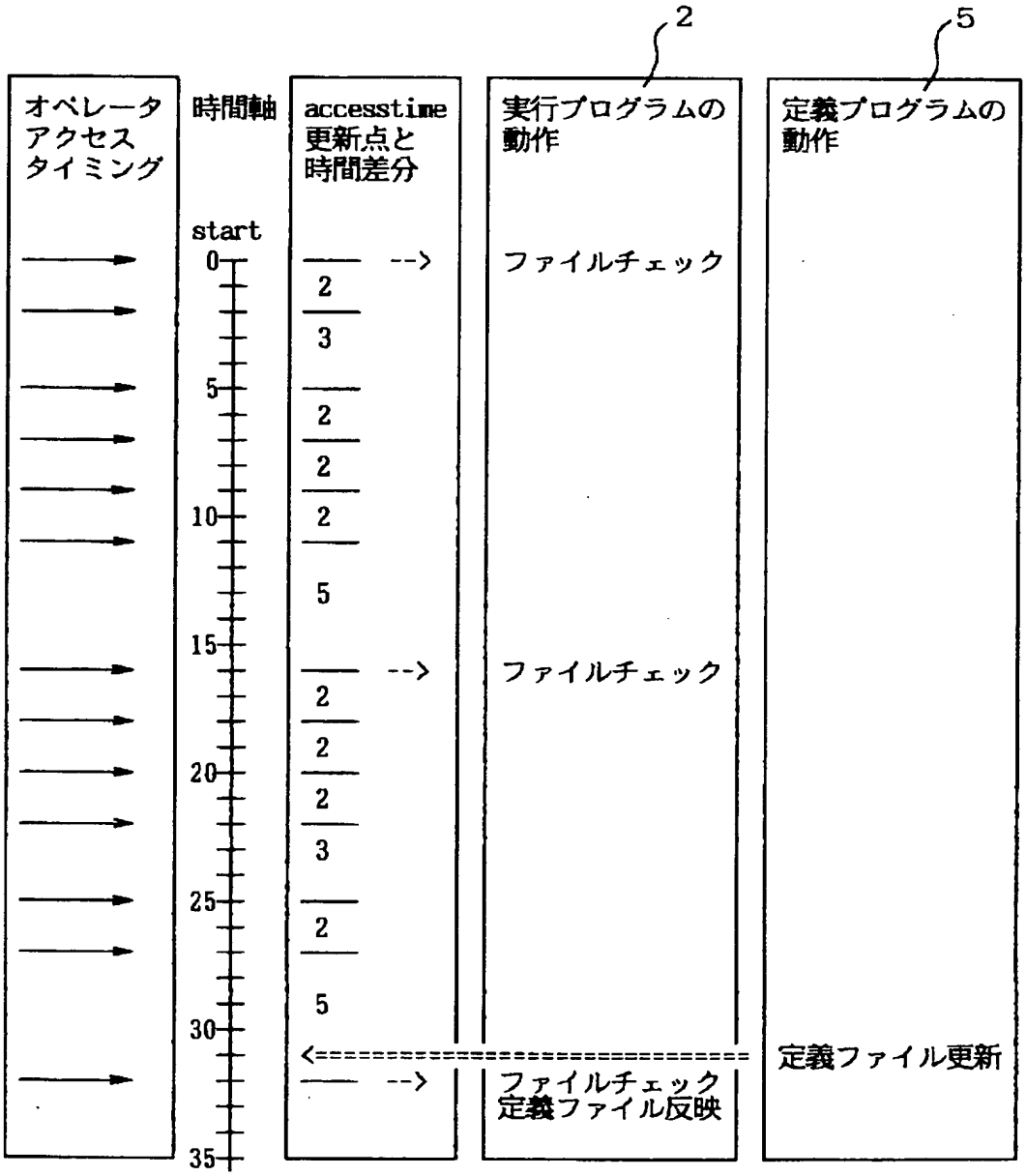
【図２】

本発明の動作環境定義ファイルの反映フローチャート



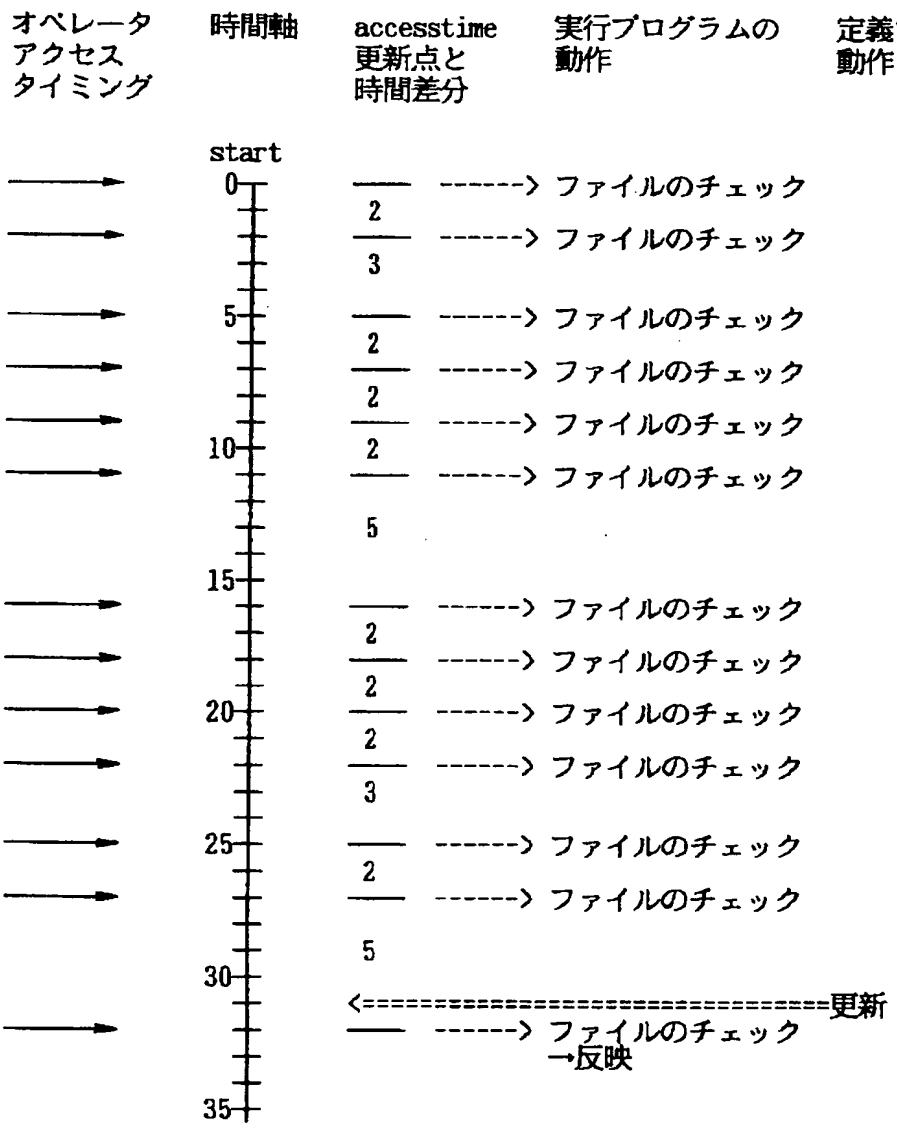
【図４】

本発明の具体例説明図



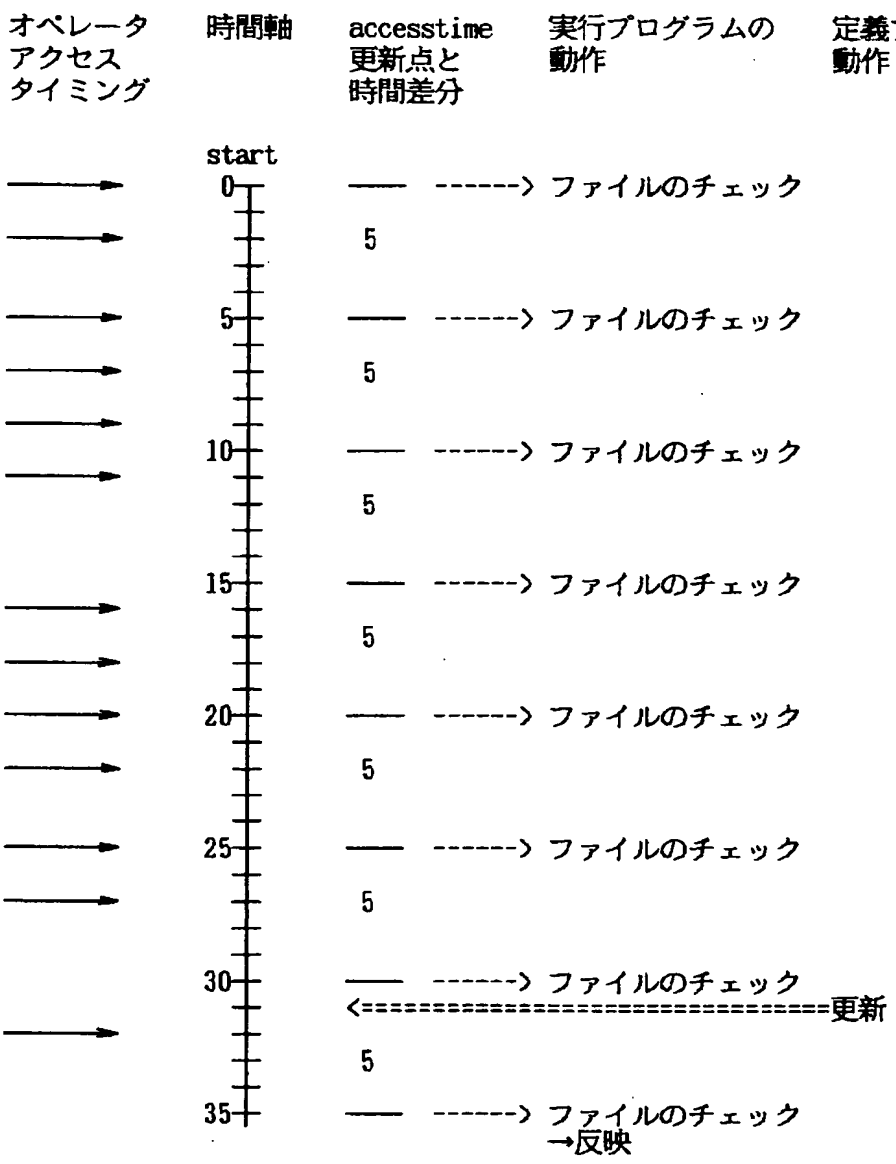
【図５】

従来技術の説明図（その１）



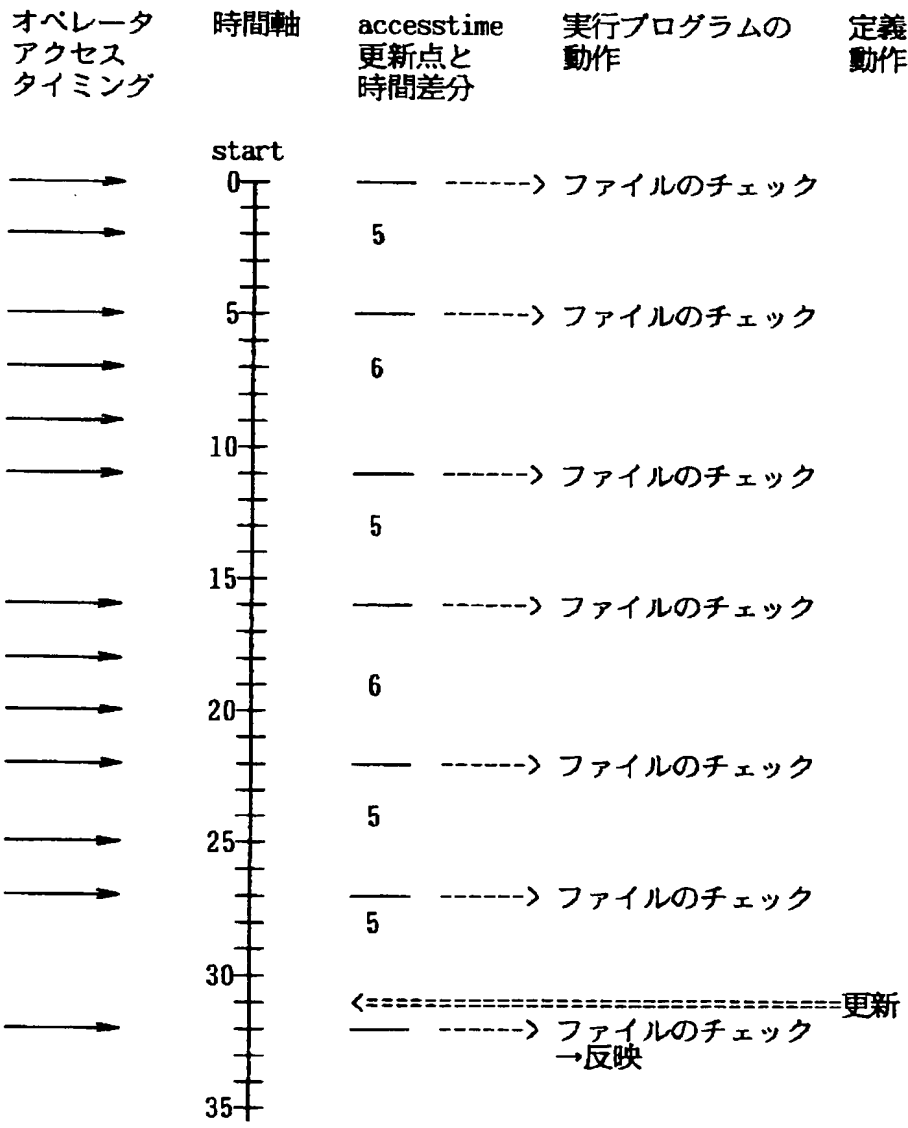
【図６】

従来技術の説明図（その２）



【図７】

従来技術の説明図（その３）



フロントページの続き

(72)発明者　渡利　亜紀子
神奈川県大和市深見西四丁目２番49号　株式会社ピーエフユー大和工場内

(72)発明者　山田　利昭
神奈川県大和市深見西四丁目２番49号　株式会社ピーエフユー大和工場内

(72)発明者　小池　匡
神奈川県大和市深見西四丁目２番49号　株式会社ピーエフユー大和工場内